RasPi

DESIGN
BUILD
CODE

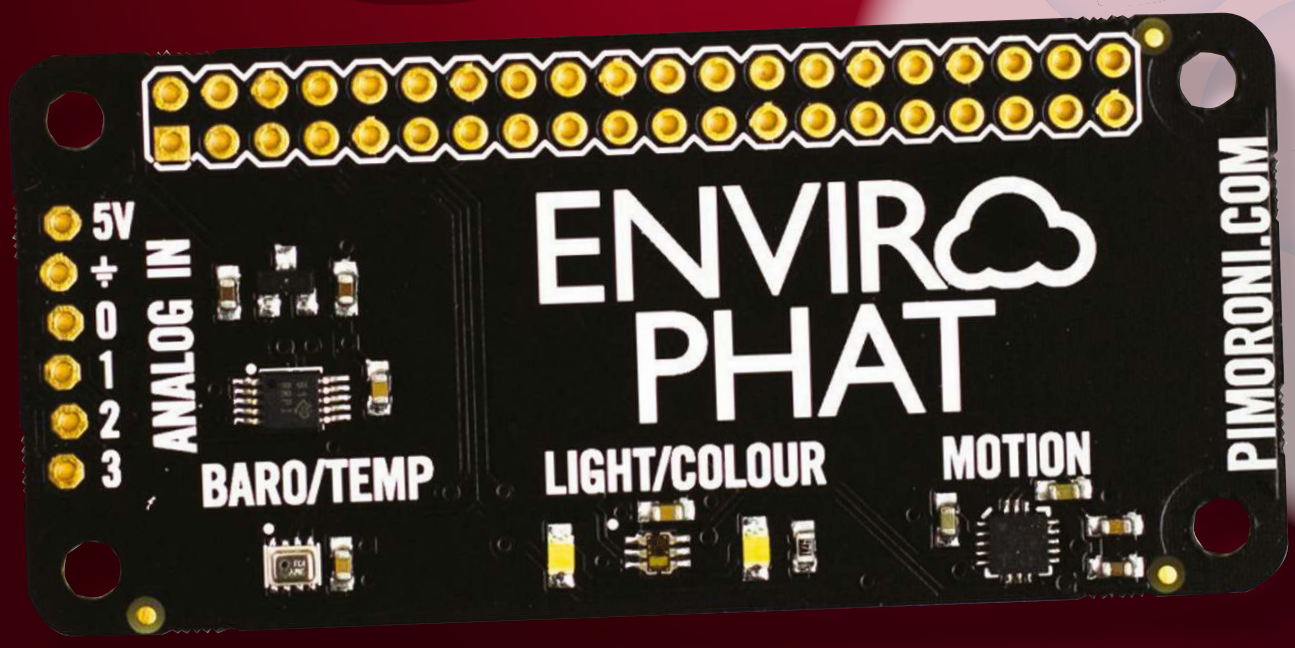
40

Get hands-on with your Raspberry Pi

BUILD YOUR OWN

GHOSTBUSTER

+7 AMAZING PI PROJECTS



Plus Encode your secrets using musical notes

Welcome

"Who you gonna call? Enviro pHat!" Would Ghostbusters have been such a huge hit in the 80s with this tagline? Probably not. It's doubtful Ray Parker Jnr would have had such a huge hit on his hands either… With Halloween disappearing into the ether we've decided to prolong the spookiness in this issue of RasPi by offering up some supernatural content of our own, kind of... With the help of our main tutorial you can learn to deploy the Pi, Enviro pHAT and an infrared camera to capture evidence of anything up to a Class 5 Full-Roaming Vapor. Alternatively you could just use it to monitor your home remotely, it's up to you. It's a lot more affordable and far less dangerous than proton packs and ghost traps anyway. Enjoy the issue!

Get inspired

Discover the RasPi community's best projects

Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

Learn to make and code gadgets with Raspberry Pi

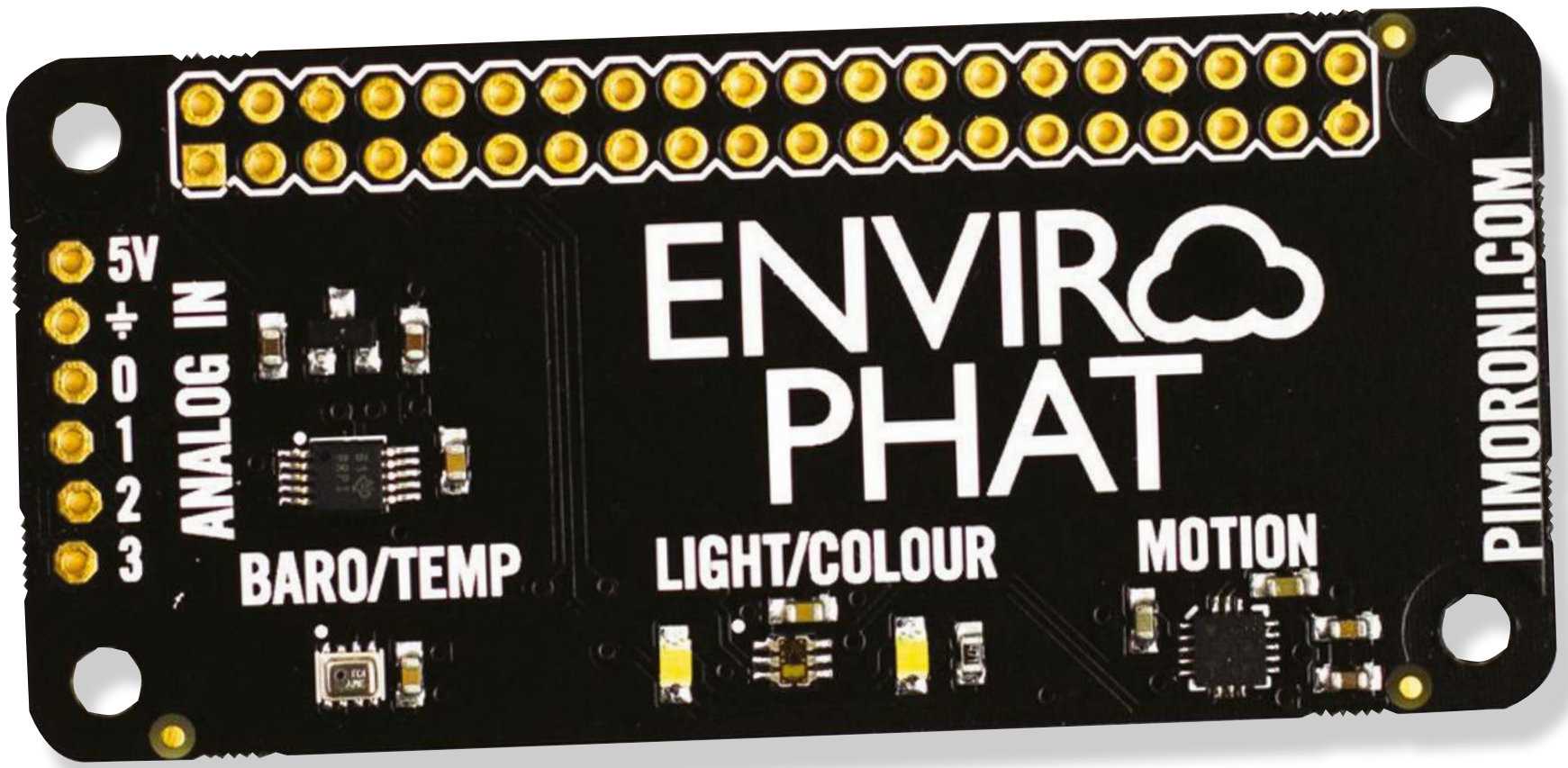
Editor

From the makers of

LinuxUser & Developer

Join the conversation at…

@linuxusermag | Linux User & Developer | linuxuser@futurenet.com



Contents

A Pi owner's guide to ghosts and how to catch them

Deploy the Pi, Enviro pHAT and an infrared camera to capture evidence of anything up to a Class 5 Full-Roaming Vapor

As a child, your author distinctly remembers learning that if a ghost or spectre is present, there are several environmental changes that can occur. First, the room may feel colder due to a sudden drop in temperature. Second, objects may mysteriously move by themselves to a new location. Third, you may observe a strange coloured light or a bright white light close to where the supernatural entity is.

Pimoroni's Enviro pHAT is the perfect hardware suite to catch ghosts. It packs four different sensors, letting you measure temperature, pressure, light level, colour, 3-axis motion, compass heading and analogue inputs. They state that "It's ideal for monitoring conditions in your house, garage or galleon. Set up a web server with Flask and remotely monitor everything from anywhere." What they failed to mention is that you can also use it to catch ghosts.

Use the BMP280 temperature/pressure sensor to measure a drop in temperature, the TCS3472 light and RGB colour sensor to pick up any changes in light, and then the LSM303D accelerometer/magnetometer sensor for movement – that book being flown across the room or the

THE PROJECT ESSENTIALS

Raspberry Pi

Enviro pHAT
http://bit.ly/EnviropHAT

LISIPARIO
www.lisiparoi.com

Black Hat Hack3r
http://bit.ly/HATHack3r

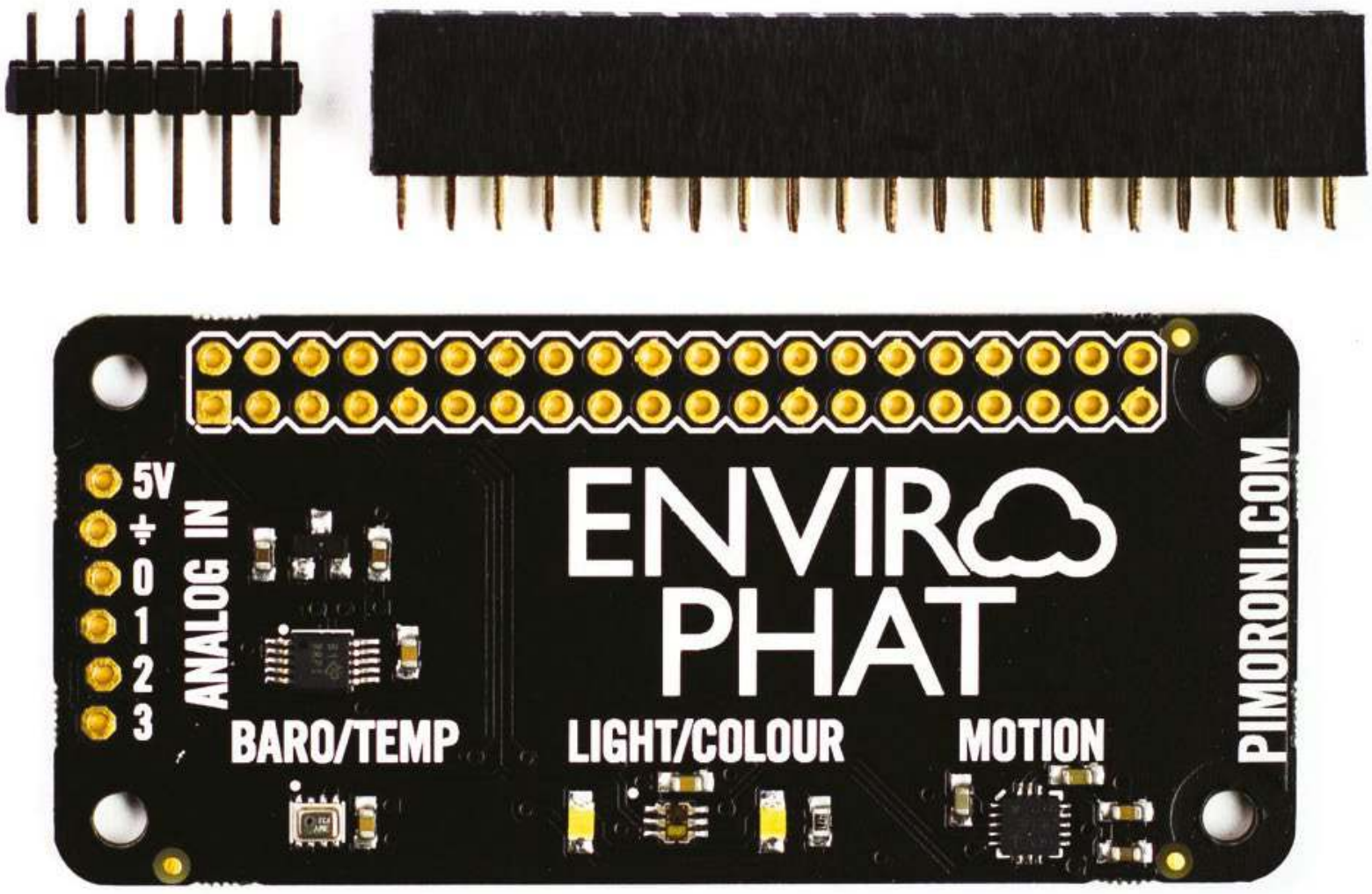


table rocking or a door opening all on its own.

Combine the Enviro pHAT With the Pi Camera Module and an infrared (IR) LED ring and you have the perfect setup to capture photographic evidence of your supernatural visitors.

01 Install the required software

Begin by setting up your Raspberry Pi and installing the required software libraries. Open up a Terminal window and update and upgrade the OS, lines one and two. Then install the Enviro pHAT library from Pimoroni by typing the command on line three. This will download and install the required software and several example programs. To make

```
pi@raspberrypi:~ $ curl -sS https://get.pimoroni.com/envirophat | bash

This script will install everything needed to use
Enviro pHAT

Always be careful when running scripts and commands copied
from the internet. Ensure they are from a trusted source.

If you want to see what this script does before running it,
you should run: 'curl https://get.pimoroni.com/envirophat'

Note: Enviro pHAT requires I2C communication

Do you wish to continue? [y/N] y

Checking environment...
Updating apt indexes...
.........E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporari
ly unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another proc
ess using it?
Apt failed to update indexes!
.....................^C
pi@raspberrypi:~ $ sudo curl -sS https://get.pimoroni.com/envirophat | bash

This script will install everything needed to use
Enviro pHAT

Always be careful when running scripts and commands copied
from the internet. Ensure they are from a trusted source.

If you want to see what this script does before running it,
you should run: 'curl https://get.pimoroni.com/envirophat'

Note: Enviro pHAT requires I2C communication

Do you wish to continue? [y/N] y

Checking environment...
Updating apt indexes...
...............
...
Checking hardware requirements...

Checking for packages required for GPIO control...
.................
```

use of an audio warning, install the MP3 library, mpg321, line four. Shut down your Pi typing the command sudo **shutdown**, then unplug the power supply.

```
  sudo apt-get update
  sudo apt-get upgrade
    sudo curl -sS https://get.pimoroni.com/
envirophat | bash
  sudo apt-get install mpg321
```

02 Set up the hardware

Next, we need to connect the hardware. It is recommended that you use the Mini Black Hat Hack3r, although the pinout website **https://pinout.xyz** provides a wiring diagram which means you don't have to. Attach the Pi Camera Module with the blue strip pointing away from the HDMI port. Next, attach the Black HAT

```
┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

P1 Camera       Enable/Disable connection to the Raspberry Pi Camera
P2 SSH          Enable/Disable remote command line access to your Pi using SSH
P3 VNC          Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI          Enable/Disable automatic loading of SPI kernel module
P5 I2C          Enable/Disable automatic loading of I2C kernel module
P6 Serial       Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire       Enable/Disable one-wire interface
P8 Remote GPIO  Enable/Disable remote access to GPIO pins

          <Select>                                  <Back>
```

Hack3r and place the Enviro pHAT onto the pins. This then provides additional pins for the wiring of the LISIPAROI IR LED ring. Last, enable the Pi Camera in the OS settings: open the Terminal window, type sudo raspi-config and select the required setting. Reboot your Raspberry Pi.

03 Wire up the LISIPAROI

The LISIPAROI requires only four wires. Attach the power wire, the one on the far right, to the 5V pin (physical pin 2). The two middle pins are used for the ground / GND connections. Any two of these physical board pins can be used: 6, 9, 14, 20, 25, 30, 34 and 39. The last connection on the left is the GPIO pin, which in this tutorial uses GPIO 10 (physical pin 19). Attach the required wires (refer to **Fig 1**)

04 Test the camera

Create a small program to test that the Pi Camera is connected, enabled and working correctly. Open your Python editor and copy out the program below. This will start a camera preview which you can view on your monitor. Then the camera will take a picture and save it onto your Pi desktop. If it does not work, check that the camera is connected correctly, that it is enabled in the Pi settings and that the code is correct.

```
from picamera import PiCamera
from time import sleep
camera = PiCamera()
camera.start_preview()
sleep(5)
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()
```

05 Transmit a signal

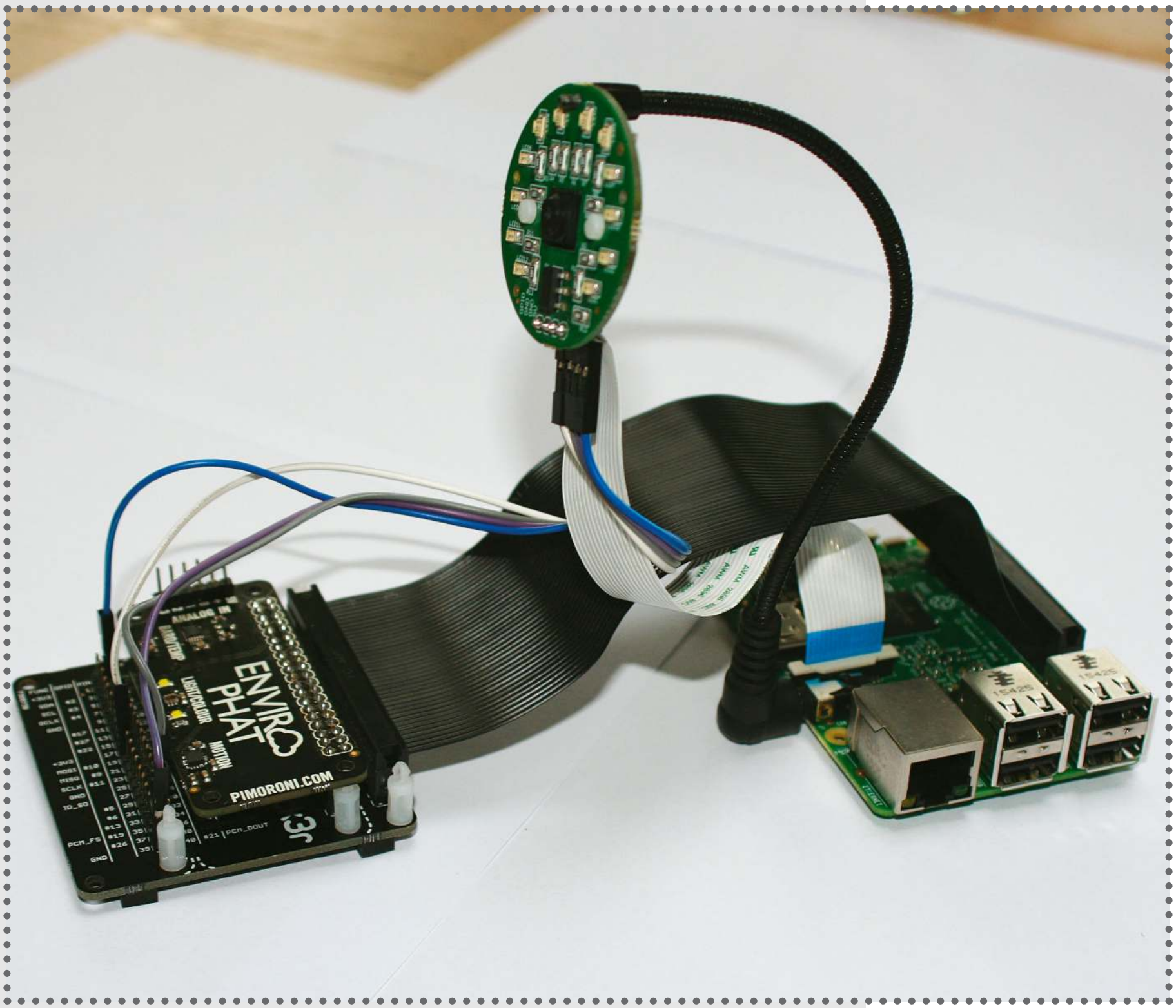
Now to create the full program. Begin this step by opening a new, blank Python file; this will hold the main program for the Ghost Detector. Import the required modules – system and OS – to write the image files back to a folder. On line four, import the Pi Camera functions and line five, the GPIO software. On the last line, import the main Enviro pHAT features: light, weather (for the temperature) and motion.

```
import sys
import os
import time
from picamera import PiCamera
import RPi.GPIO as GPIO
```

```
    from envirophat import light, weather,
motion, leds, analog
  camera = PiCamera()
```

06 Set values for the motion sensor

Set the GPIO pin numbering system to BCM; this option means that you are referencing the pins by the 'Broadcom SoC channel' number, line one. These are the standard numbers after the 'GPIO' label, instead of the physical pin number. Set GPIO 10 as the output, line two;



this is used as a trigger for the IR LEDs. Next, create three variables to hold the required values for the motion sensor. The first value refers to the amount of movement to detect, the second is a list to hold the gathered sensor readings, and the third is the last position of the measurement on the z axis.

```
  GPIO.setmode(GPIO.BCM)
  GPIO.setup(10, GPIO.OUT)

  #variables for the motion sensing#
  threshold = 0.2
  readings = []
  last_z = 0
```

07 Save images and take a sensor reading

If you are lucky enough to encounter a spectre then you will have photographic evidence. This file is named and saved as a picture number. Create a global variable to store the picture number, line one. If you capture more than one image, you do not want to overwrite the previous one. Set the initial image value at 0, line two.

```
  global file_name
  file_name = “0”
```

08 Create a camera function – part 1

The main program works by checking a reading from a sensor and if it falls within a range of values then it triggers a function which controls the camera. Begin by naming the function, line one. Then add the global filename, line two; this means you can increment the

filename each time a new image is taken. One line three, set the GPIO 10 to HIGH. This creates a circuit on pin 10 and sends power to the IR LEDs, turning them on, resulting in the camera being able to take an image in the dark.

```
  def catch_a_ghost():
    global file_name
    GPIO.output(10, GPIO.HIGH)
    time.sleep(0.2)
    print (file_name)
```

09 Create a camera function – part 2

Complete the function by triggering the camera

Catch_a_ghost.py - F:\Freelance Work\Linux Tutorials\Ghost Catcher 1st September\Catch_a_ghost.py (3.6.0)

File Edit Format Run Options Window Help

```
#!/usr/bin/env python

import sys
import os
import time
from picamera import PiCamera
import RPi.GPIO as GPIO
from envirophat import light, weather, motion, leds, analog
camera = PiCamera()

GPIO.setmode(GPIO.BCM)
GPIO.setup(10, GPIO.OUT)

#variables for the motion sensing#
threshold = 0.2
readings = []
last_z = 0

global file_name
file_name = "0"

#set up a function for the camera#
def catch_a_ghost():
        global file_name
        GPIO.output(10, GPIO.HIGH)
        time.sleep(0.2)
        print (file_name)
        camera.capture(file_name + ".jpg")
        time.sleep(0.3)
        GPIO.output(10, GPIO.LOW)
        file_name = int(file_name) + int(1)
        file_name = str(file_name)
        os.system('mpg321 /home/pi/wegotone.mp3 @')
```

and setting the name of the image file, line one. Then turn off the infrared LEDs by setting GPIO 10 to LOW, line three. To avoid overwriting the image file, increment the filename by one using the code file_name = int(file_name) + int(1). The int converts the value to an integer, so that it can be added to. However, filenames are not integers, so you must convert the name back into a string, line five. The final line triggers an optional audio alarm informing you that the camera has taken a picture. You may want to leave this line out if your setup is in a remote haunted house or, if you feel tempted to check the image straight away and you don't want to risk running into the ghost. It might be best to wait until the morning.

```
  camera.capture(file_name + ".jpg")
  time.sleep(0.3)
  GPIO.output(10, GPIO.LOW)
  file_name = int(file_name) + int(1)
  file_name = str(file_name)
  os.system('mpg321 /home/pi/wegotone.mp3 @')
```

10 Establish the current room temperature

To discover if a ghost is present, you can check for a sudden change in temperature. First, we record the starting room temperature, with weather.temperature(), storing this into a variable, line one. Next, create a loop, line five, which will continually check for changes. On line six, create a new variable called current_temp and take another new reading. This value can then be used and compared with the start temperature in Step 14, to calculate if there is a significant shift in temperature.

```
  start_temp = weather.temperature()
```

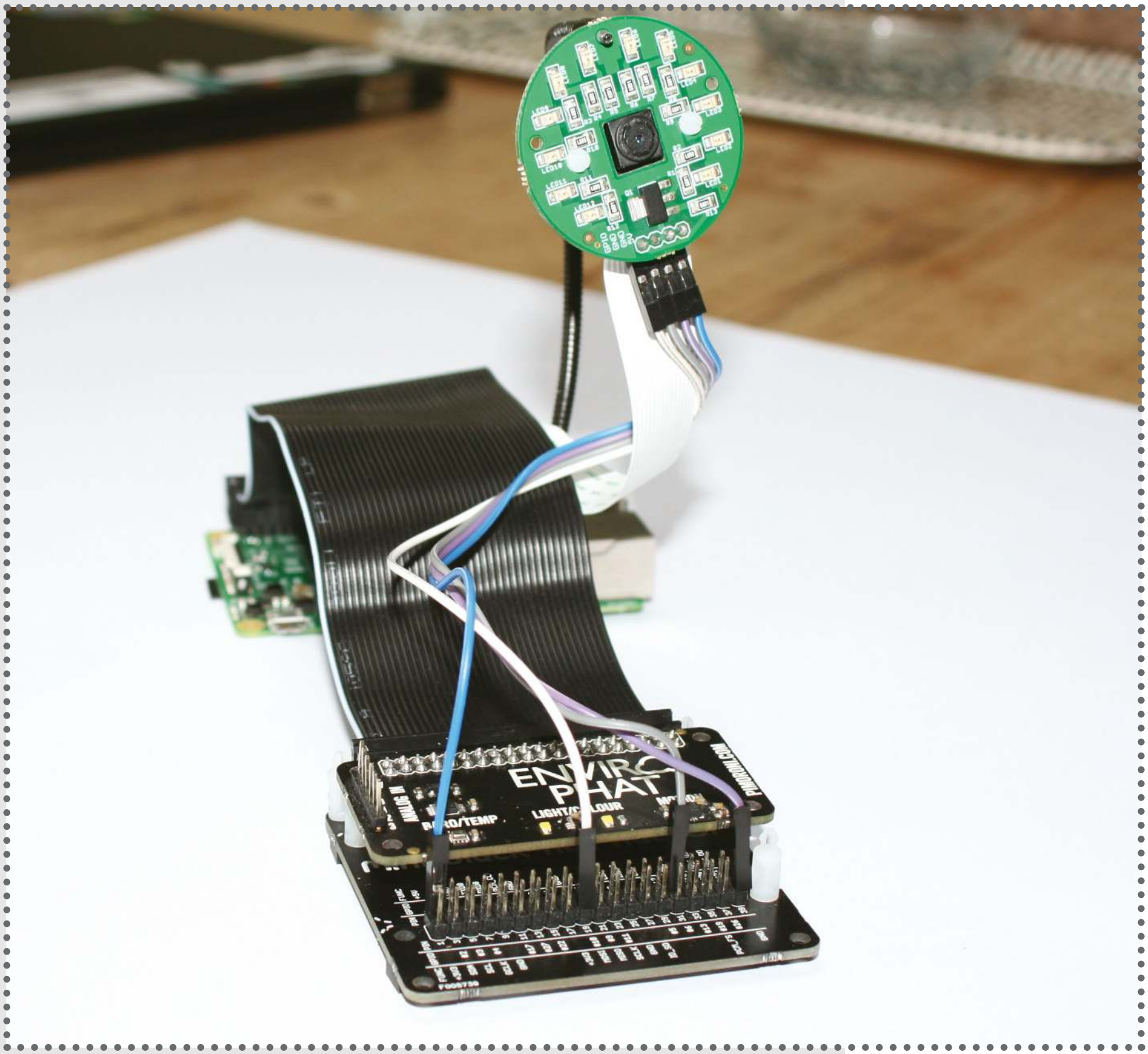
What is cron?

Cron is used as a time-based job scheduler that permits you to schedule jobs (commands or shell scripts) to run periodically at certain times or dates. It is commonly employed to automate system maintenance programs such as disk or administration tasks.

```
  print ("The room is ", start_temp)
  try:
    while True:
    current_temp = weather.temperature()
    print ("Current Temperature is", current_temp)
```

Create a new bespoke lircd configuration file –

```
irrecord -d /dev/lirc0 ~/lircd.conf
```



Steampunk Laptop

Robbie Frazelle went for the wow-factor with a birthday gift, a Pi-powered steampunk-themed laptop

Where did the idea for the steampunk laptop come from? I hear it was gift for your girlfriend.

For our first ever Christmas I made her an ornate mirror with the magic mirror platform (https://magicmirror.builders). Super-cool. I was wandering through this thrift store and found this giant ornate frame for 20 bucks and I thought that's it, I'm going to make it for her [...] Immediately after comes her birthday. Mid last year she'd spilled coffee on her MacBook so we talked about a back-up computer.

That was my mentality around putting together some kind of laptop. I don't really know where the idea of making it steampunk came from. If I could tell you how my head works, I'd probably freak out a lot of people. Aesthetically, it just seemed to make sense. When you're building a wooden laptop, there's already that steampunk direction, right?

Robbie Frazelle is director of marketing for an interactive lighting company. Prior to that, he worked as a designer for Optiv Security (formerly FishNet Security), a cybersecurity company.

We were intrigued due to the details you added, like the marble lighting.

The whole idea wasn't baked when I started building. I knew I was going to do a box of a laptop. As I started building that out and piecing things together, the idea continued to evolve. When I figured out my power supply had LEDs on the front, I thought it would be really nice to see those LEDs on the front of the box. The initial thought was to rewire LEDs to that board inside the power supply and have those sticking up at the front. The connects on that LED board on that PCB board were tiny. There was no possible way for me to do that. I tried and it didn't work [laughs].

That's when I thought about running some fibre optic lines and then thinking about how I could diffuse that

light. In the town where I live there's a glass blower, so I approached him with the project. That worked out for that part. Every little aspect of that box started as a question: I kind of I want to do this – how do I do that?

What were the significant issues that you encountered?

So I started the whole thing off with the keyboard and the keyboard tray. I knew I wanted a typerwriter-esque keyboard and [...] a vintage typewriter is not cheap. That's not going to work, then I found a mechanical keyboard typewriter set and I could see that was the way to go, so I started building this keyboard tray. The initial tray I built was actually two steps back from the actual [one] that went into the box. I didn't know how the lid was going to close, so putting in the support on the side, that wasn't cut out right the first time. At first I thought I was going to put that battery monitor on the top of the keyboard tray – but that changed.

Have you found the Pi works OK, as the computer within the computer?

For the intent for this specific computer, yes. Because all she needs it to do is some word processing, so LibreOffice and check her email and browse the internet. The Raspberry Pi 3 does that fairly well [...] Now, if this were the Raspberry Pi 2 or even the B+ or the A, browsing the internet is like pulling teeth. I felt like I was on 386DX, on a dial-up connection waiting for things to pop up. With the advent of the Raspberry Pi 3, this made it more of a reality. Even now it still has its issues, it's not as speedy as it could be, but I have full faith in Raspberry Pi that they will continue to

THE PROJECT ESSENTIALS

- Oak planks and trim
- Recycled laptop LCD
- Talentcell rechargeable Li-ion battery
- AC rocker switch
- Magnetic catches
- 3PDT switch
- 3mm fibre-optic cable
- 1× HDMI cable
- ½-inch Braided PET sleeving
- Bronze gears
- Raspberry Pi case with fan

upgrade their hardware.

Are you gearing up for Christmas for the next project, now you've set this bar so high for yourself?

I set the bar, right. If I'm not getting calls for interviews from the UK for her next project then I failed, I think [laughs]. I don't know what the next project is going to be. I build arcade cabinets as well. I've been using RetroPie [https://retropie.org.uk] to do these really elaborate arcade machines with a very simplistic hardware and software mentality to them. I'm building one right now.

Below The end result is beautiful, but Frazelle admits that some more preparation may have avoided a number of rethinks and setbacks, particularly with the keyboard tray.

We saw the Zelda arcade machine and it's beautiful work.

That was my first iteration, I've done five since then and they have consistently gotten better. The biggest improvement was switching from plywood to MDF. That's a game changer and then doing things like adding T-moulding to the sides. When you put the T-moulding on at the end, all of a sudden it goes to awesome from ridiculously basic.

Like it?

As mentioned in the interview, this is just one of many projects that Frazelle has undertaken using the Pi, including one that turns an ornate mirror into a smart device using the magicmirror2 platform **(https://magicmirror.builders)**, which includes facial recognition, traffic and weather information.

Do you see RetroPie as the best emulation software to use for this?

I've not played with a lot of other ones [...]. I operate on the mentality of if it ain't broke then why fix it and the RetroPie front-end works for everything I need it to, even skinning the front-end. I've been designing my own themes for each of my arcade cabinets that fit in with the overall design of the project. It gives me that flexibility and it runs the emulators that I need it to. That being said, I've been looking at attract-mode [http://attractmode.org] recently. It has this really nice scroll wheel on the side of games and puts an arcade picture next to them and when you flip through them it shows animated screens of the videogames as you go through them. Super-cool from a visual aspect. It's not completely stable yet, so I've not jumped over to it. I think it runs with RetroPie, but it replaces Emulation Station with a different front [...]. I'm all about the eye candy, I like it when it looks neat. I want people to sit down and go "Whoa, that looks really cool!" That's what I'm shooting for. As long as it's got the bells and whistles and the crazy lights.

Further reading

We had to reduce the list of components as there are so many, but Frazelle details the whole build process at **http://bit.ly/** SteamPunkLaptop. Many of his other projects are frankly breathtaking and the Legend of Zelda bartop arcade cabinet is a particular labour of love: http://bit.ly/ ZeldaArcade Cabinet.

Raspberry Pi doomsday switch

Cement your Bond villain status using a Raspberry Pi and the handy Pibrella board and create a switch of Doom

Any evil genius worth their salt knows that a key part of their arsenal, besides a secret base inside a volcano, is a doomsday device which when triggered will release nuclear launch codes. Not all of us have aspirations of world domination, but we can appreciate the usefulness of a doomsday device which can execute commands at a moment's notice. In this project, we'll be using the versatile Pibrella add-on board (available for just £10 from http://bit.ly/Pibrella) to build your very own device.

Aside from having the requisite big red button, the board also comes with a handy buzzer which can be programmed to signal a countdown when you activate the doomsday switch. The Pibrella also has three built-in LEDs (green, yellow and red), which we'll also use.

Once the switch is pressed, a special doomsday Python script is launched. We've included some code samples to allow you to send an email or erase data on your Pi, which you can configure using the guide below. But we encourage you to customise it as you see fit.

THE PROJECT ESSENTIALS

Raspberry Pi

Pibrella

http://pibrella.com

01 Prepare your Pi

Using any model of Pi with GPIO pins, you'll need a clean install of the latest version of Raspbian – to make sure it's up to date, open a Terminal window and run sudo apt-get update then sudo apt-get upgrade. Attach the Pi to your router via Ethernet or use the network manager to

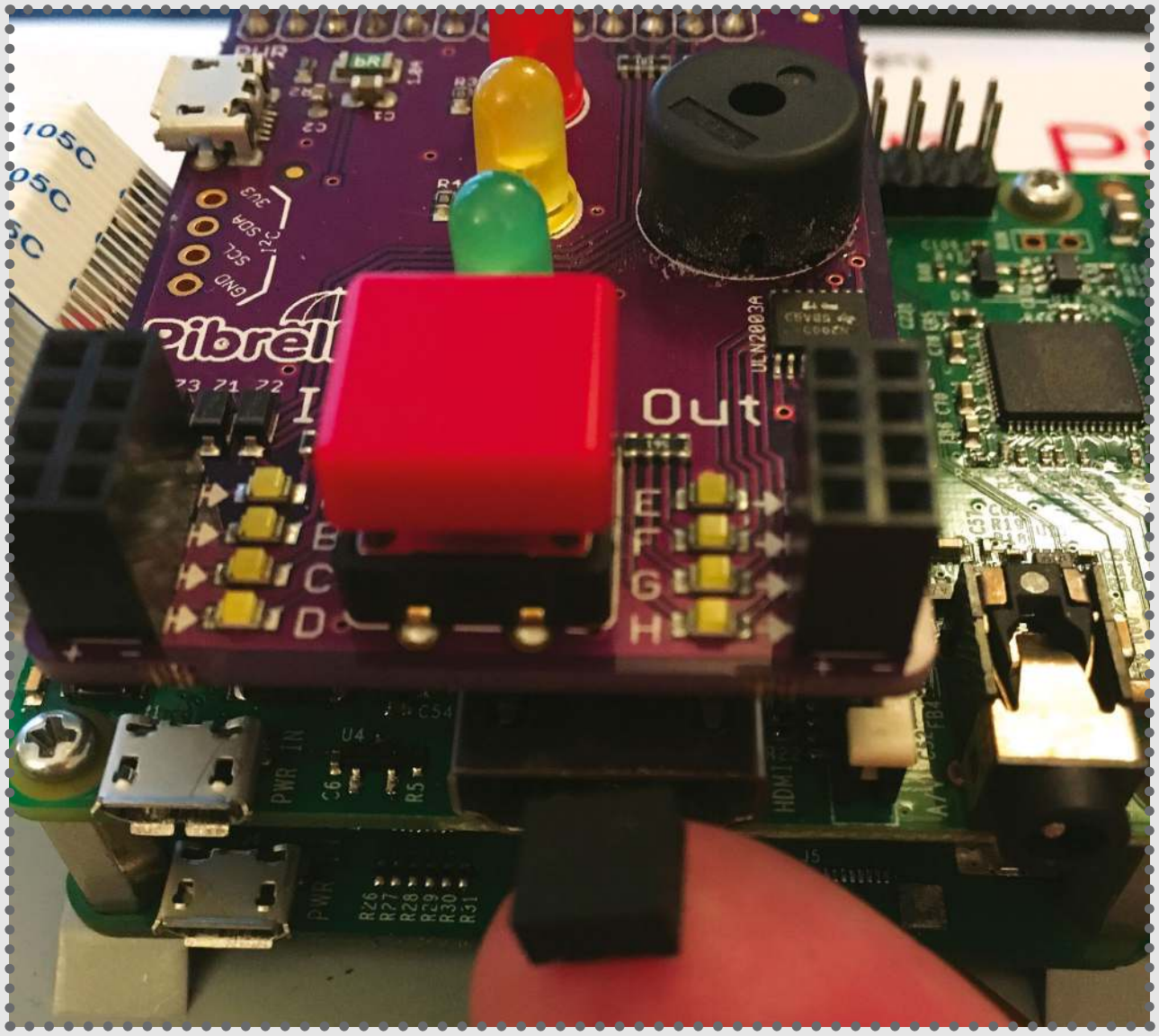
connect to your local wireless network.

02 Connect your Pibrella

Remove the Pibrella from its anti-static bag and connect it to your Pi's GPIO pins. If you're using a Raspberry Pi 3, you may notice that there are more pins than corresponding connectors on the Pibrella. The Pibrella must be connected to first 26 GPIO pins – this is the end furthest from the USB ports (see photo above to check correct positioning)

03 Set up Pibrella

If the Pibrella is connected correctly, you should see a small blue LED light up on the corner of the board. If not, check that it's mounted on the GPIO pins properly and try again. The board also comes with a small adhesive



square of sponge which you can place between the Pibrella and the Pi's HDMI port, to stop it bending when you press the red button.

Next, open a Terminal on your Raspberry Pi and enter curl -sS get.pimoroni.com/pibrella | bash to run the Pibrella installer. Press Y to continue. I2C will download and install. Next, press Y once again to perform a full install.

04 Configure Pibrella

After the installer has downloaded the necessary files, run the command sudo python -i to create your first Pibrella python script.

Enter the text import pibrella. Press Return to start a new line, then enter pibrella.light.pulse(0.2). This should cause all three LEDS on the Pibrella board to start blinking.

Once you're happy that the Pibrella is working, use the command quit() to exit. The Pibrella will also generate some text to confirm it has shut down cleanly.

Before proceeding any further, use Terminal to download your very first Pibrella script by running wget https://raw.githubusercontent.com/nate-drake/pibrella-doomsdaydevice/master/isconnected.py. There's also a copy of this script on this month's cover disc and on FileSilo.

05 Configure the 'is connected' script

In Terminal, run sudo nano isconnected.py. This opens the script inside the nano text editor.

As you can see (in the screenshot at the bottom of the previous column), the script imports the pibrella module once again, as well as one named urllib2, which

```
#!/usr/bin/env python

import urllib2

import pibrella

import socket
REMOTE_SERVER = "www.google.com"
def is_connected():
  try:
    # see if we can resolve the host name -- tells us if there is
    # a DNS listening
    host = socket.gethostbyname(REMOTE_SERVER)
    # connect to the host -- tells us if the host is actually
    # reachable
    s = socket.create_connection((host, 80), 2)
    return True
  except:
     pass
                                [ Read 23 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

– you've guessed it– can request HTTP URLs.

The module will try to resolve the web, determined by the value under REMOTE_SERVER. By default this is www.google.com, but feel free to change this if your doomsday switch is dependent on a particular service running, such as your email provider.

Press Ctrl+X, Y, then Enter to save and exit. Test the script with sudo python /home/pi/isconnected.py if you wish.

06 Configure 'primed' script

The primed.py script is programmed to wait until you press the red button on the Pibrella, at which point it will launch the doomsday script itself. Like the isconnected.py script, it runs in a continuous loop and illuminates the yellow LED while doing so. This is an excellent way to tell that the doomsday device is ready, therefore 'primed'.

Download the script by running the command wget https://raw.githubusercontent.com/nate-drake/pibrella-doomsdaydevice/master/primed.py. Next, run nano primed.py to examine the script.

The primed.py script uses the os Python module to launch the doomsday.py script once the button is pressed. The global variable doomsday prevents

```
###END OF DOOMSDAY SCRIPTS####


 Define countdown parameters. By default this is 10.0 seconds.

 = Timer(10.0, activate)

 Start the countdown
.start()

If the red button is pressed during the countdown, the program will exit.

hile countdownover ==0:
                pibrella.buzzer.note(2)
                pibrella.light.red.blink(1, 1)
                if pibrella.button.read() == 1:
                                t.cancel()
                                sys.exit()
```

doomsday.py being relaunched if the button is pressed more than once.

07 Configure doomsday timer

If you haven't done so already, press Ctrl+X, Y, then Enter to save and exit the primed.py script.

Next, download the doomsday script by running wget https://raw.githubusercontent.com/nate-drake/pibrella-doomsdaydevice/master/doomsday.py.

Use the command sudo nano doomsday.py to enter the text editor once again.

Scroll down to the section marked '# Define countdown parameters'. Any lines that begin with a '#' are ignored by Python, so they're used here to explain what each section of code does.

Feel free to amend t = Timer(10.0, activate) to an interval of your choice by amending the text '10.0'; e.g. t = Timer(3.0, activate).

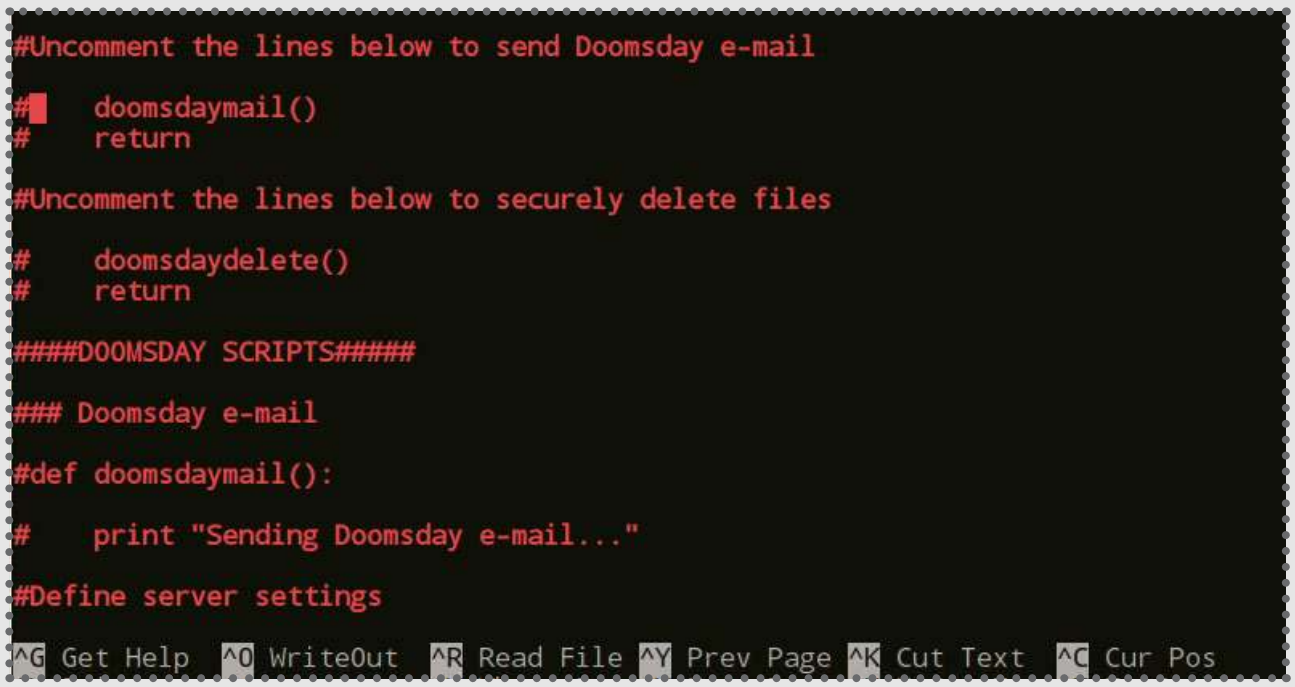
08 Choose doomsday commands

Scroll up to the section of code marked 'def activate():'. The activate function runs after countdown is complete. By default, all this does is switch off the buzzer and print a message saying 'Countdown is Over'. The red LED will also stop blinking and remain steady.

The doomsday script can be configured to do anything you wish, as long as it's programatically possible in Python. To get you started we've included two functions, one of which will send an email and another which will delete files or folders on the Pi itself.

To enable these functions, remove the '#' at the start of the relevant lines (see below).

09 Enable email and choose server



Find the section marked #Uncomment the lines below to send Doomsday e-mail, then remove the '#' at the start of the next two lines. Next, scroll down to the section marked ### Doomsday e-mail. Remove the '#' from the start of the two lines reading:

```
#def doomsdaymail():
#    print "Sending Doomsday e-mail...
```

Indentations and spaces are very important in Python code, so don't change the text formatting.
Remove the '#' at the start of the line marked # server = smtplib.SMTP('smtp.gmail.com', 587). This defines the

SMTP server and port used to send your mail. Feel free to change this if you use a different provider.

10 Configure doomsday email settings

Find the section marked # Define sender and receiver e-mail address. Remove the '#' at the start of the two lines:

```
#   sender = 'youremail@address.com'
#   receiver = ['receiversemail@address.com']
```

Change the values of youremail@address.com and receiversemail@address.com to your own email address and that of the recipient respectively.
Next, find the section marked #Request TLS connection. Remove the # at the start of the three lines below if you connect to your mail server by TLS.
Immediately below this section, you'll find the #Login details: Remove the '#' at the start of the line reading # server.login and change youremail@address.com and password123 to your own email address and password.

11 Configure email message

```
efine sender and receiver e-mail address
 sender = 'youremaile@address.com'
 receiver = ['receiversemail@address.com']

w log in to your mail server. Change the e-mail address and password
cording to your account settings. You may want to have a dedicated e-mail
ddress for this.

quest TLS connection
 server.ehlo()
 server.starttls()
 server.ehlo()
gin details:
 server.login("youremail@address.com", "password123")"
 #Send the message
 msg = "The Doomsday Switch has been tripped! Send me an emergency donut!"
 server.sendmail(sender, receiver, msg)
 print "Closing connection..."
 server.close()
 pibrella.buzzer.success()
```

Find the section marked # #Send the message and remove the '#' from the start of the four lines below. The value msg contains some default text informing your recipient that the doomsday device has been activated and to send you an emergency donut. You're welcome to edit this as you see fit.

You can optionally also remove the '#' from the line # pibrella.buzzer.success() to have the device play a special sound once the email has been sent. This is an excellent way to make sure that the doomsday device has worked without connecting it to a monitor.

12 Enable doomsday delete

If you want your doomsday device to delete files on the Pi, find the section marked #Uncomment the lines below to securely delete files and remove the '#' from the start of the two lines below. Scroll down to the section marked ### Start Doomsday Delete and remove the '#' from the start of these lines:

```
#def doomsdaydelete():
#    print "Attempting to delete files.."
```

```
#    pibrella.buzzer.success()

### End of Doomsday e-mail


### Start Doomsday Delete

def doomsdaydelete():

    print "Attempting to delete files..."

#Delete a folder securely including its contents
#The file is overwritten 38 times. Last pass is all zeroes.

#    folderpath = "/home/pi/Documents/private"
#    subprocess.call(["srm", "-rvz", folderpath])

#Delete a file securely using 'shred'
#The file is overwritten 25 times. Last pass is all zeroes.

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Doomsday trouble

If you run into any trouble with your doomsday script, restart your Pi, then try to run it from Terminal with the command sudo python doomsday.py. Use print after each command to locate issues; e.g. print "Connecting to server...".

This will activate the 'doomsday delete' function. The Pi has no built-in function to securely erase folders, so before proceeding, click File > New Tab, then run the command sudo apt-get install secure-delete.

13 Securely erase folders

Find the following lines in doomsday.py that are marked:

```
# folderpath = "/home/pi/Documents/private"
# subprocess.call(["srm", "-rvz", folderpath])
```

Remove the '#' at the start of these to uncomment them. The value folderpath determines which folder to erase. By default this is an imaginary folder named private, but you can change this to any folder you wish.

The doomsday delete makes use of the subprocess module here, which, like os.system in the primed.py script, can be used to run system commands.

The advantage of using subprocess is that you can use command-line options. Here, the srm utility is configured to securely erase folders and their contents.

14 Securely erase files

If you just want to erase individual files, uncomment the lines marked:

```
#Delete a folder securely including its contents
#The file is overwritten 38 times. Last pass is all zeroes.

    folderpath = "/home/pi/Pictures/vacation"
    subprocess.call(["srm", "-rvz", folderpath])

#Delete a file securely using 'shred'
#The file is overwritten 25 times. Last pass is all zeroes.

#    filepath = "/home/pi/Documents/secretfile1.txt"
#    subprocess.call(["shred", "-zu", filepath])
#    print "Files have been shredded."
#If files are deleted, the buzzer will make the 'success' sound.

#    pibrella.buzzer.success()

### End Doomsday Delete
```

```
    #          filepath   =   “/home/pi/Documents/
secretfile1.txt”
    #          subprocess.call([“shred”,    “-zu”,
filepath])
```

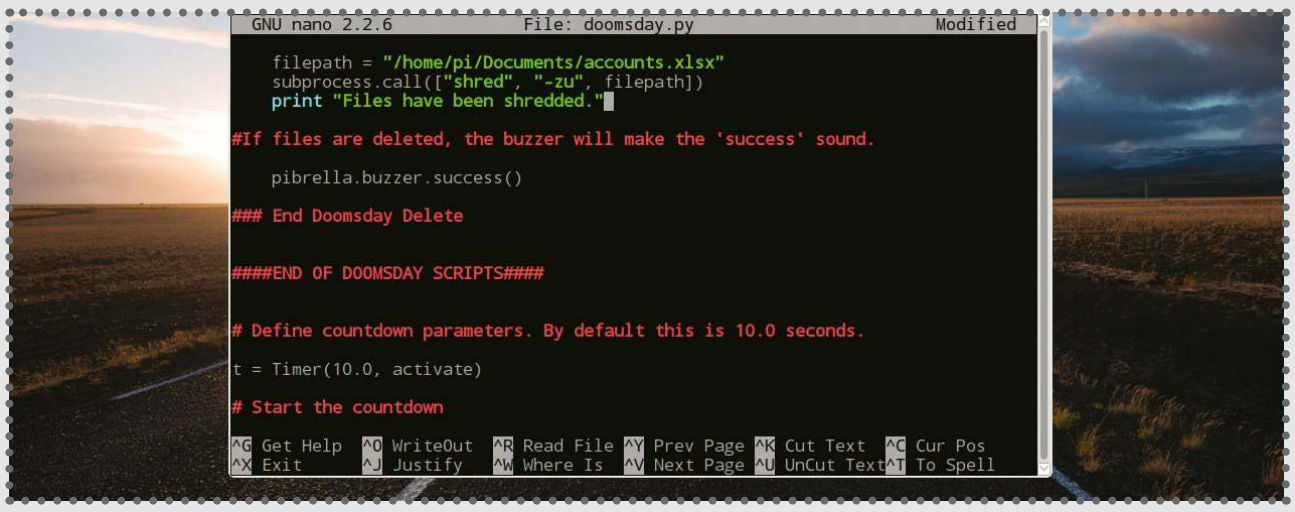
The filepath value operates in a similar way to folderpath in the last step, in that it specifies the exact location of certain files. Feel free to amend the filename and path as you see fit.

If you have more than one file to delete, simply copy and paste these two lines with a new filepath; for instance:

```
              filepath   =   “/home/pi/Documents/
secretfile1.txt”
               subprocess.call([“shred”,    “-zu”,
filepath])

      filepath = “/home/pi/Pictures/secretfile2.
jpg”
                    subprocess.call([“shred”,    “-zu”,
filepath])
```

Uncomment ‘# pibrella.buzzer.success()’ to make the



Pibrella play a sound once the files have been deleted.

15 Configure startup

Press Ctrl+X, Y, then Return to save and exit the doomsday script. Your next step is to make sure your new doomsday device is always active.

In Terminal, run the command sudo nano /etc/rc.local. This script determines which programs start up when you log in to your Pi. By default it does nothing. Make sure there's a '#' at the start of every line except the one that reads 'exit 0'. Above this line, paste the following:

```
sleep 30
python /home/pi/isconnected.py &
python /home/pi/primed.py &
```

Next, run chmod +x /etc/rc.local to make the script executable, then reboot your Pi.

16 Test the doomsday device

Once you've logged back into your Pi, wait for 30 seconds for the green and yellow LEDs to light up. A brief reminder at this stage – the isconnected.py script checks to see if you're connected to the internet and blinks green if successful. This means if your doomsday device requires connectivity, you'll know everything is in order. The yellow LED should be steady, indicating that the primed.py script is waiting for you to press the button. Do so now. The buzzer will sound and the red LED will blink. Once the countdown is over, the doomsday.py script will try to launch the commands you configured.

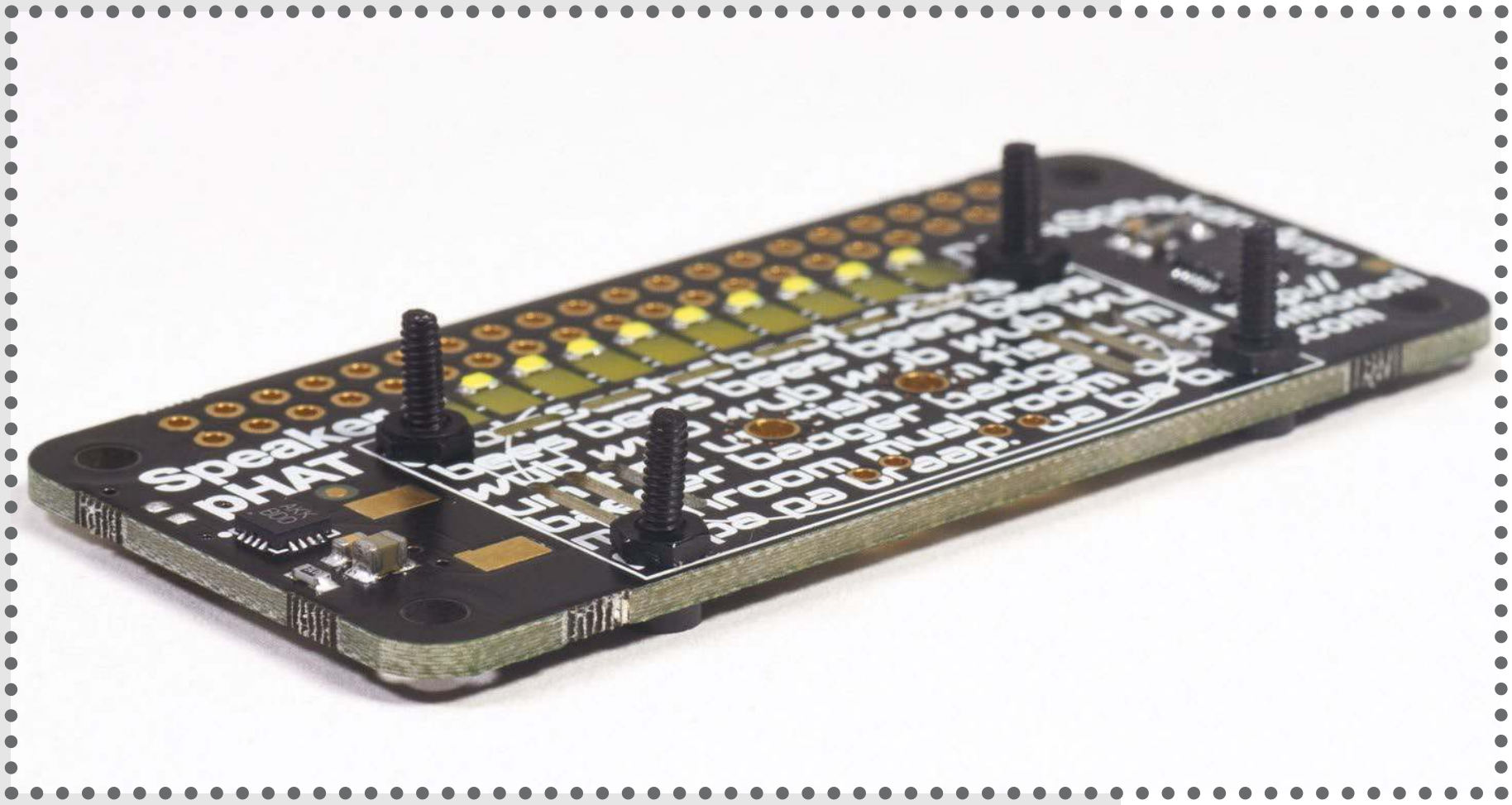
Getting started with the Pimoroni Speaker pHAT

Add audio to your Raspberry Pi via the Speaker pHAT. Set up the hardware, install the libraries and bang out some tunes

The Raspberry Pi Zero is awesome. However, it lacks an 'on the go' audio solution. Worry no longer, as Pimoroni has created an '80s-style boombox speaker known as the Speaker pHAT. It crams an I2S DAC and mono amplifier, a tiny 8 2W speaker and a 10 LED bar graph onto one teeny little pHAT. A simple solution to add audio to your Pi, it is compatible with all models.

Now you can integrate music, notifications, speech and sound into all your projects. It comes as a complete kit, and this tutorial walks you through how to set up and prepare the hardware; the installation of the software; how to alter and adjust the volume setting; and finally, for those who want to, how to customise and control the LED indicator.

Because the hardware is designed specifically with the Raspberry Pi Zero in mind, there are a number of elements to consider before you get started. First, the DAC sums the two audio channels to mono, rather than just outputting one channel or the other, meaning that you get both channels of audio out of the single speaker.



Second, the dimensions of an assembled Speaker pHAT are H 10mm x L 65mm x W 30mm; this means that it will not fit on top of a Pibow Coupé or Pibow Zero case due to the height of the speaker. Finally, the install knocks out the HDMI audio and basically routes all audio through the pHAT speaker.

01 Solder the board

Start by soldering the GPIO header onto the Speaker pHAT. Ensure that the header is kept securely in place. You can use some masking tape – don't use plastic tape! – to hold it while you solder the first couple of pins. Begin by anchoring the corner pins; you can then remove the tape. Solder the two opposite corner pins to anchor the header in place, before soldering the rest of the pins.

02 Go solderless

If your soldering isn't up to standard, you can add the

required pins using 'hammer headers'. These come in both male and female versions and use 'crafty little retaining nubbins' to grip tightly into the holes on the PCB. The kit comes complete with an installation jig that contains two acrylic base pieces. Sit your Pi into the jig and secure with the two nylon bolts, placing the acrylic top piece onto which you hammer. A more detailed guide can be found at http://bit.ly/HammerHeaders.

03 Mount the speaker

Now we'll add the speaker to the pHAT using the four screw holes. Use four of the supplied eight nuts as spacers to keep the speaker slightly away from the PCB. Place each screw through the hole and then add a nut. Then place the speaker on the screws and secure with the four remaining nuts.

04 Solder the speaker wires

Once the speaker is securely in place, you need to solder the speaker wire contacts to the board. The kit comes with the required wire; simply measure a suitable length and cut it. Using the soldering iron, heat the solder on top of the speaker. When the solder is melted, attach the wire. Solder the bottom of the wire to the Speaker pHAT base. Repeat this process for the second wire.

05 Install the software

As with all Pimoroni products and software, it's super-easy to install the required libraries and configuration files. Attach the Speaker pHAT to your Raspberry Pi, add a power supply and boot it up. (Do not add the pHAT while the Pi is turned on). Open the Terminal and simply type curl -sS https://get.pimoroni.com/speakerphat

Stream music from iPhone or iPad

It is possible to stream music via your iPhone or iPad to the Pi and Speaker pHAT. Assuming you've already installed the pHAT software, you will need to use a clean SD card image. Install the required libraries: type the command curl -sS get.pimoroni.com/airdac | bash, then select Speaker pHAT. The required libraries and configuration will be set up. Last, it will install the shairport-sync software that enables you to stream music via AirPlay to your Raspberry Pi.

| bash. This will begin the installation process. Follow the displayed instructions, answering as required. Remember that the installation will knock out the standard HDMI audio. You can re-enter the command in the future to update the software if a new edition is released.

06 Playing audio

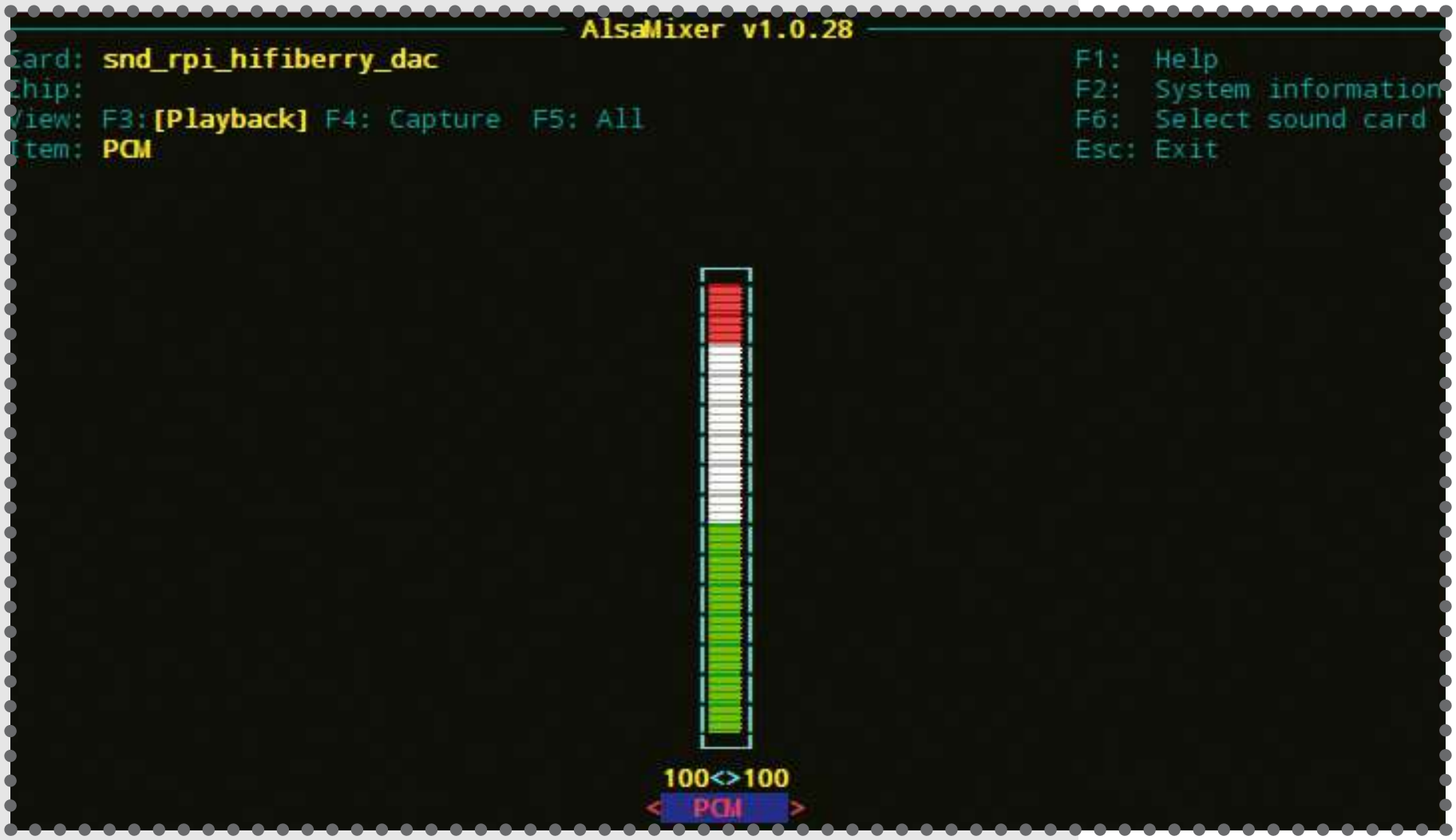
Once the installation process has completed, it will prompt you to restart your Raspberry Pi. This is recommended; press Enter on your keyboard or type sudo reboot. Your Pi will restart. Now all audio will be played through the Speaker pHAT. To test this, simply open the Chromium browser and go to YouTube, or another website of your choice that has audio. Select a suitable video or song and press play. The video will load, play and the sound will be heard through the Speaker pHAT.

07 Adjusting the volume

The installation of the Speaker pHAT libraries will knock out the HDMI/analogue controls and the volume icon from the top menu bar. You may however want to adjust the playback volume; the simplest method is to use the alsamixer. Open the Terminal and type alsamixer. This opens the sound card configuration tool. You'll see that the sound card is named rpi hifiberry_dac. Adjust the volume by using the Up arrow key to increase it and the Down arrow key to decrease it until you reach the desired volume level.

08 Playing MP3s

To play MP3 files, we'll use a program called mpg321. Open a Terminal window, then type sudo apt-get install



mpg321. This will download and install the MP3 player. MP3s can now be played from the command line; if you wish to use Python, however, open IDLE 3.

09 Using Python to play MP3s

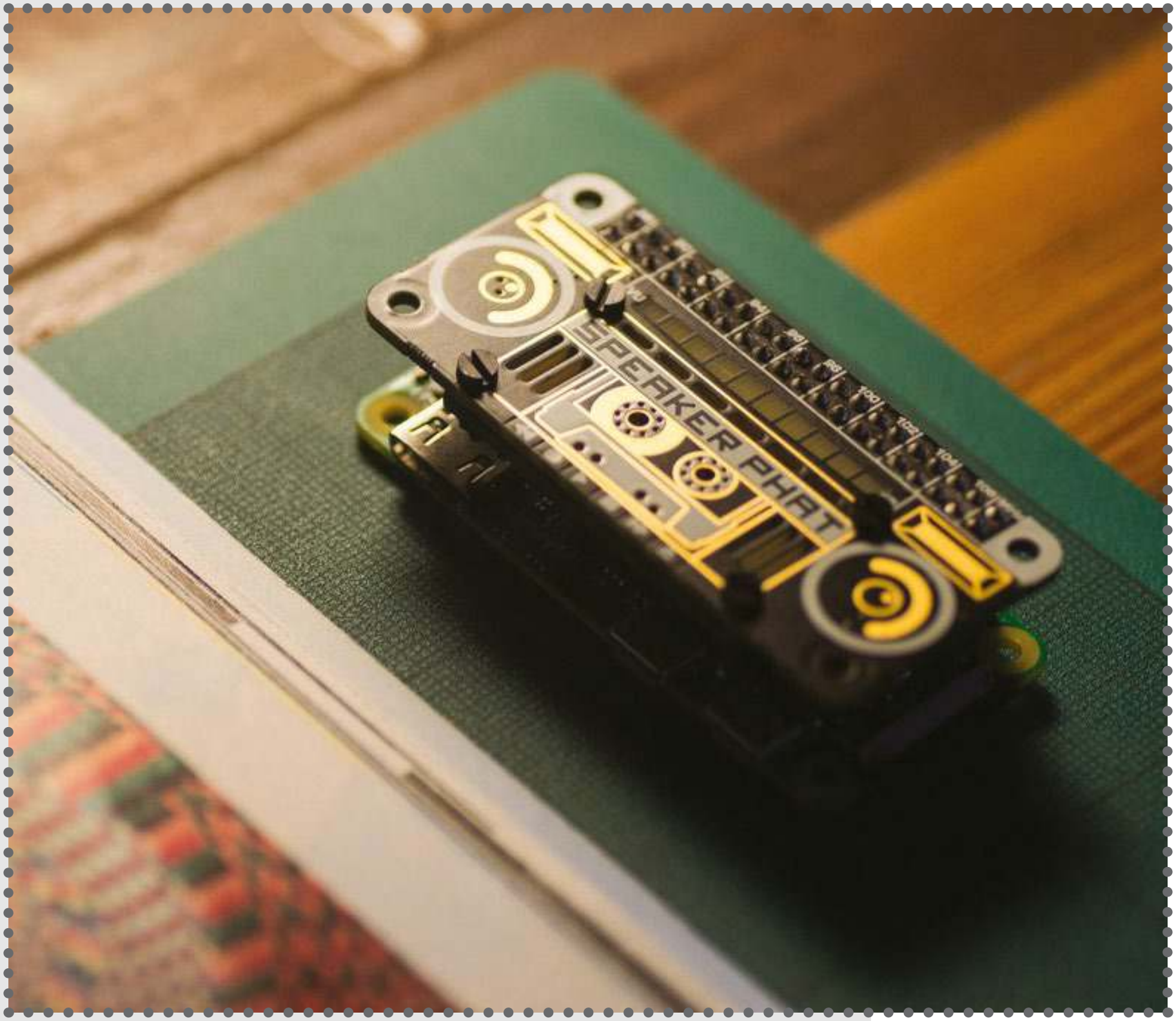
Begin by creating a New File, and then add the code import os to the top line of the program. On the second line type os.system ('mpg321, name_of_mp3_file.mp3'). This invokes an OS command to use the mpg321 player. Replace name_of_mp3_file.mp3 with the name of your desired MP3 file to be played. Press F5 to save the program and play the music file.

10 Editing the LEDs

It is possible to control the Speaker pHAT's LEDs. First, install the Python sn3218 module: sudo apt-get install python-sn3218 python3-sn3218. Create a new Python file and enter the following code.

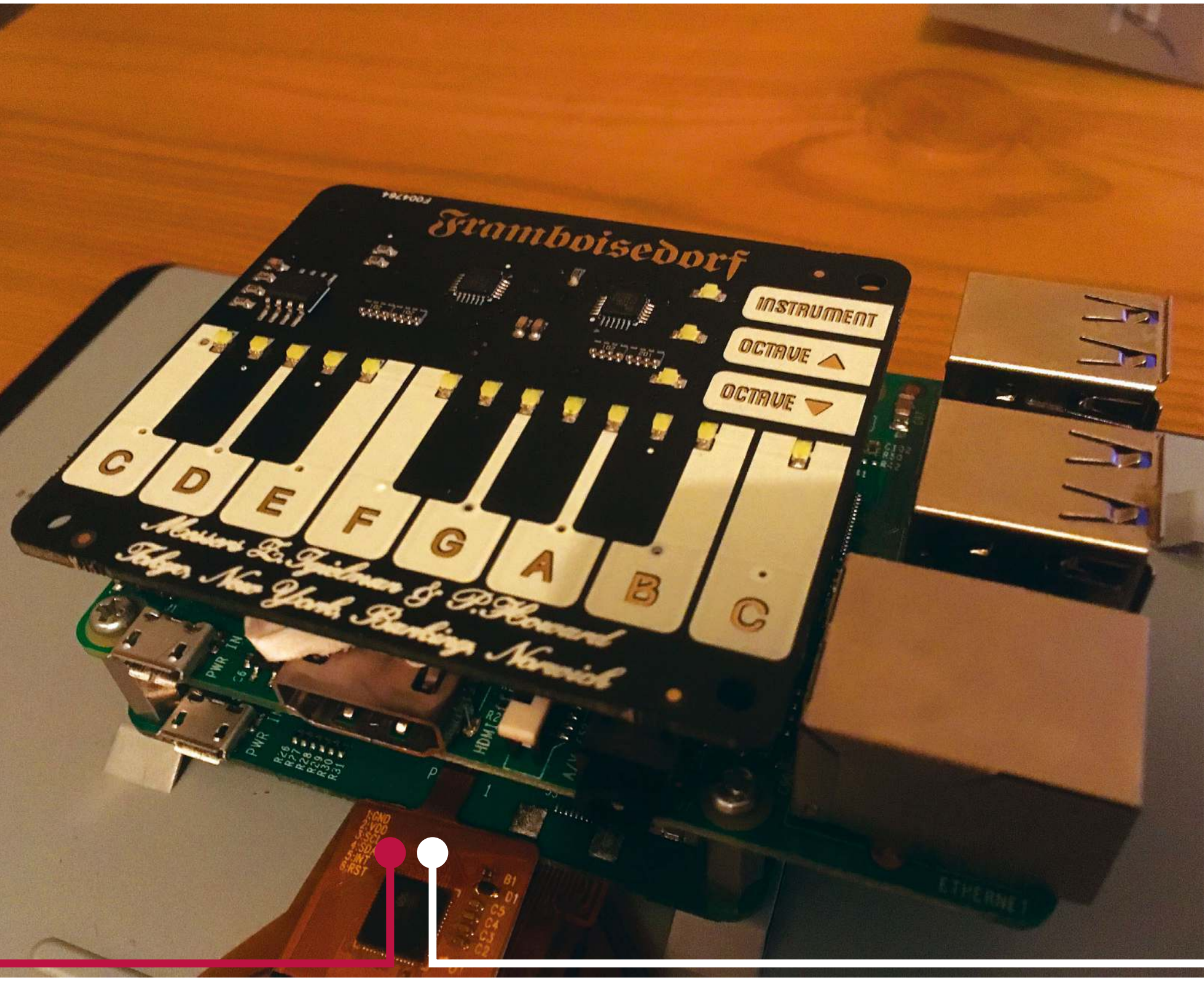
```
from speakerphat import clear, show, set_led
clear()
for x in range(10):
  set_led(x,255)
show()
```

For the set_led(index, value) function, index can be 0-9; value 0-255. Save and run the program.



Make musical passwords

Mozart meets Moonraker with a Pi project for musically encoding your secrets using Pimoroni's Piano HAT



The Pimoroni Piano HAT is an extraordinarily useful board that can turn your Raspberry Pi into an electronic keyboard. The keys are responsive to touch and each one has an LED which can light up as you play. Pimoroni has thoughtfully compiled an excellent library of coding samples for your Pi which enable you to crank out tunes within minutes of receiving your Piano HAT.

In this project we will talk you through the basics of attaching the Piano HAT to your Pi, as well as how to use it to generate 8-bit tones and play back piano and drum sounds. You will then learn to use the custom 'pianohat-password' Python scripts to encode a secret message using musical notes. The first script, piano-password.py, allows you to record a musical password by playing a few notes, then invites you to enter some text you wish to hide, such as a password or email address. Use the second script, **piano-decode.py**, to decode your hidden secret.

01 Set up your Pi and add Piano HAT

This guide requires a clean install of the latest version of Raspbian on your Pi. The easiest way to ensure this is to visit http://bit.ly/Pi-NOOBS and follow the steps to install NOOBS to your microSD card. Once the installation is complete, run sudo apt-get update and sudo apt-get upgrade to bring Raspbian fully up to date.

The Piano HAT itself is very easy to install. Marry up the Pi's GPIO pins to the connector on the HAT. The Piano HAT is compatible with all models of Raspberry Pi; however, if you have a Pi Zero or Zero W, you will need to attach a male GPIO header yourself.

02 Download installation scripts

Open a Terminal window on your Pi, or connect via SSH, and run the command:

```
curl https://get.pimoroni.com/pianohat | bash
```

Read the message about I2C and press Y to continue. The installer will then ask if you wish to install the Piano HAT samples. Press Y to confirm you wish to do this too, as you can use these to check the Piano HAT is working later.

Installing the samples also downloads some of the Python modules you will need to use the Piano scripts, such as NumPy.

If successful, the installer will display the message: 'All done, enjoy your Piano HAT!'

03 Test 8-bit synth keyboard

If you chose to download the Piano HAT samples in the previous step, there is now a folder named Pimoroni in

```
pi@raspberrypi:~ $ sudo python /home/pi/Pimoroni/pianohat/examples/8bit-synt

8-bit Piano HAT

This advanced example demonstrates software synthesis.

It uses pygame's sndarray.make_sound method to create
8bit tones with specific frequency, bitrate and samplerates.

Instrument = Toggle Sine Wave
Octave ^ = Toggle Saw Wave
Octave V = Toggle Square Wave

Please wait while Pygame is set up and samples are generated...

Press CTRL+C to exit!

Initializing subsystem...
Generating samples...
```

MD5 hash

Cryptographically inclined readers may have noticed that the piano password script generates an MD5 hash of the musical password. This doesn't improve security but does mean that the password is exactly 16 bytes long, which is required for AES encryption.

your home folder containing some example scripts you can work with.

Connect a speaker or a pair of earphones to the Pi. Next, using Terminal, run the following command to transform your Pi into an 8-bit synthesizer:

```
sudo python /home/pi/Pimoroni/pianohat/examples/8bit-synth.py
```

Press a few keys to 'make beautiful music'. These tones are generated using the Pygame module, specifically the make_sound method to create 8-bit tones with their own frequency, bitrate and sample rates.

04 Test simple piano and drums

When you tire of tinny 8-bit tones, press Ctrl+C and run another example Piano script:

```
sudo python /home/pi/Pimoroni/pianohat/examples/simple-piano.py
```

This script uses a series of WAV files to play a series of piano notes or drum beats (use the 'Instrument' key to

```
File  Edit  Tabs  Help
pi@raspberrypi:~ $ sudo python3 /home/pi/Pimoroni/pianohat/examples/simple-pian
.py

This example gives you a simple, ready-to-play instrument which uses .wav files

For it to work, you must place directories of wav files in:

/home/pi/Pimoroni/pianohat/examples/sounds

We've supplied a piano and drums for you to get started with!

Press CTRL+C to exit.

Loading Samples from: /home/pi/Pimoroni/pianohat/examples/sounds/drums
Playing Sound: /home/pi/Pimoroni/pianohat/examples/sounds/drums/clap.wav
Playing Sound: /home/pi/Pimoroni/pianohat/examples/sounds/drums/hat.wav
Playing Sound: /home/pi/Pimoroni/pianohat/examples/sounds/drums/hit.wav
Playing Sound: /home/pi/Pimoroni/pianohat/examples/sounds/drums/rim.wav
Playing Sound: /home/pi/Pimoroni/pianohat/examples/sounds/drums/smash.wav
Playing Sound: /home/pi/Pimoroni/pianohat/examples/sounds/drums/thud.wav
Playing Sound: /home/pi/Pimoroni/pianohat/examples/sounds/drums/ting.wav
Playing Sound: /home/pi/Pimoroni/pianohat/examples/sounds/drums/ting.wav
```

switch between the two).
Use Ctrl+C to quit the script, then locate it in the file browser. Right-click to open it with Thonny IDE.
As you will see, the script runs a procedure each time one of the 16 buttons on the Piano HAT keyboard is pressed. Piano keys are numbered according to note and octave. For instance, the first octave of C is numbered 24.

05 Download pianohat-passwor

Now that you're satisfied the Piano HAT is working correctly and understand its basic workings, it's time to download pianohat-password, which is a modified version of the example script simple-piano.py.
Open a Terminal on your Pi and run:

```
git clone https://github.com/nate-drake/pianohat-password.git
```

Don't forget to place the sounds folder in the same folder as the scripts in order to hear tones as you play.
Before proceeding, run the following commands to install the required software for encryption:

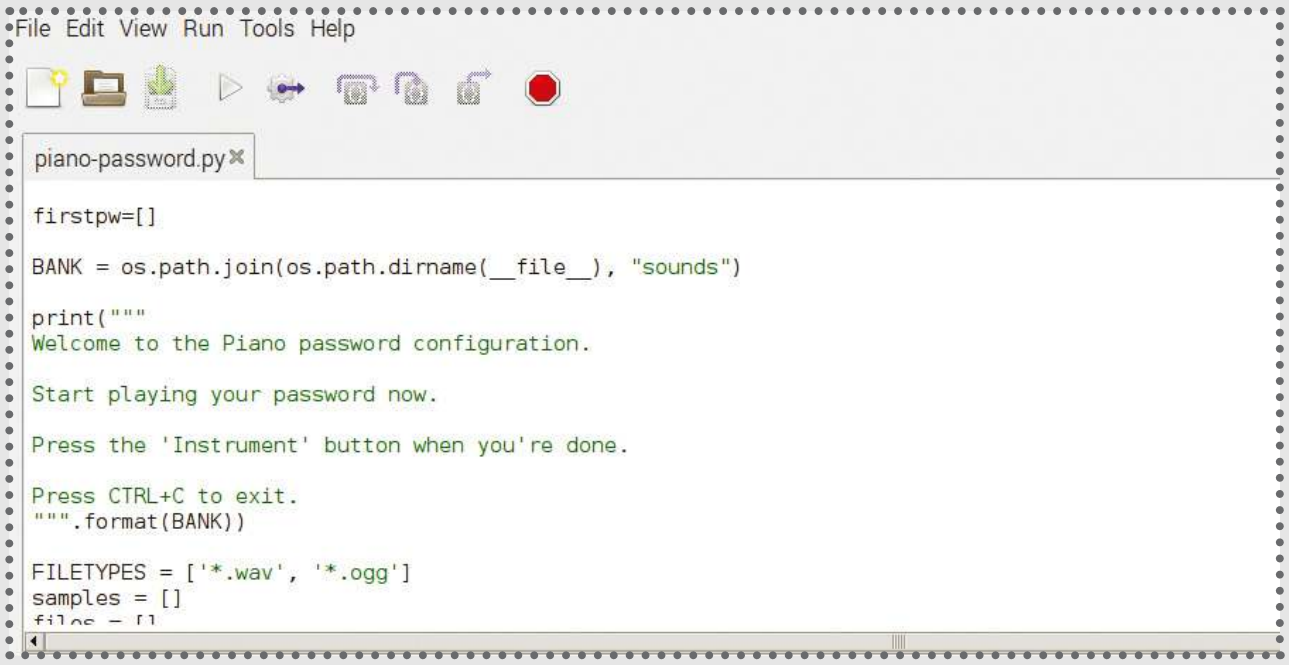
```
sudo apt-get install build-essential libffi-dev
sudo pip install bcrypt pcrypto
```

06 Review password entry

Open your File Explorer and navigate to the new folder named pianohat-password. Right-click on the script

piano-password.py, then open it using either the Thonny IDE or the Text Editor.

When the script runs, it asks you to play the notes that will appear on your keyboard. The 'Instrument' button now launches the procedure confirm_password.



The value firstpw is a list, which is used provisionally to store the password as you play. Scroll down to the procedure handle_note. Each time a key is pressed, the relevant number (channel) is displayed and this value is added to the list firstpw.

07 Test simple piano and drums

Next, scroll down to the procedure named confirm_password. This procedure is launched when the user presses the 'Instrument' button to submit the password. In the first instance, the script will display the numeric equivalents to the keys. The for loop is used to play back the tones via the speaker every 0.7 seconds, to ensure you remember them.

If you are serious about security, you may well prefer to use a # to comment out these lines, since it's generally a bad idea security-wise to show your password in

'plain' text.

The password is then converted from a list (firstpw) to a string (strpw), in preparation for it being hashed.

08 Download pianohat-passwor

Once the key tone password has been converted to a string, the piano-password script generates an MD5 hash of it. This MD5 hash in turn is then converted to yet another hash using bcrypt. The advantage of doing things this way is that bcrypt automatically uses a unique 'salt' when encoding passwords. This makes it highly unlikely that anyone who chooses the same musical password as you will have exactly the same hash.

This also means, however, that each bcrypt hash is unique, so the script saves it to a hidden file named .pianohash. Feel free to change the filename and location if you wish.

09 Review password entry

Once the password has been saved, the piano-password script launches the encode_text procedure. This makes use of the PyCrypto module. The raw_input value (now simply input in Python 3) prompts you as the user to enter your secret data.

The encryption algorithm used is AES, which works on 16-byte (128-bit) blocks of data. The encode_text procedure uses a while loop to pad out the text. If you wish to change the scheme used, you may need to edit this.

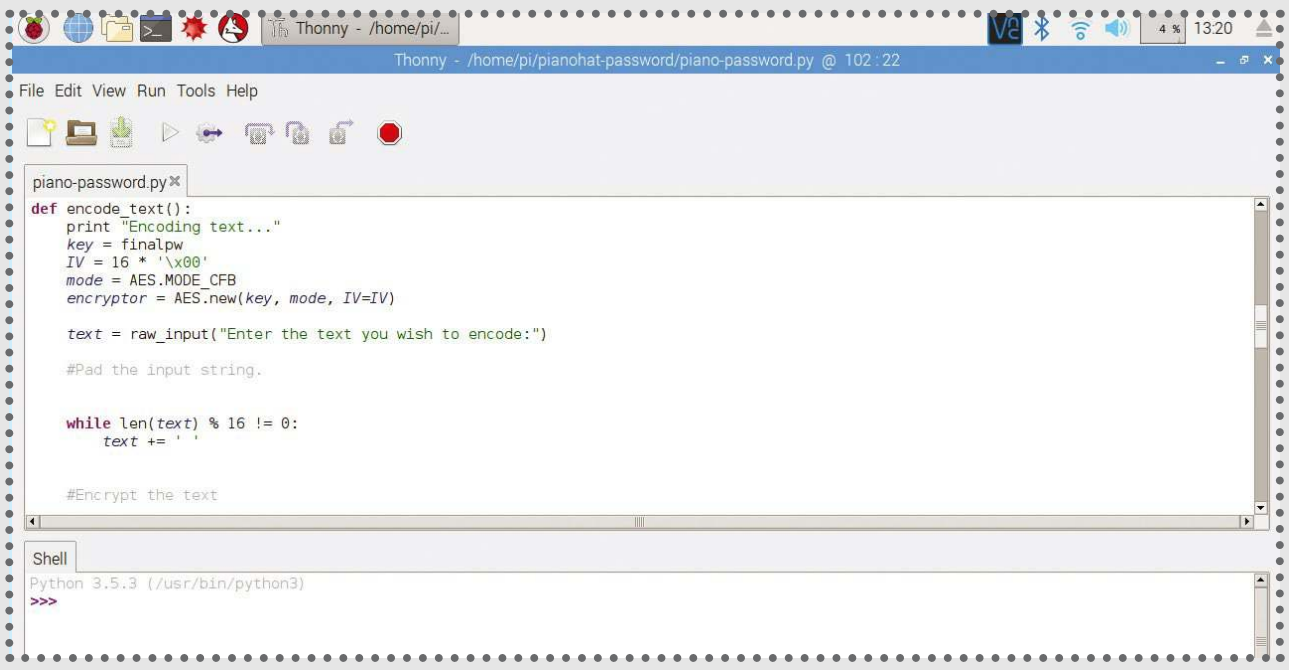
The text you input is saved to a hidden file named .pianovault in the same folder as the piano-password

script. Feel free to change this if you wish.

10 Review password confirmation

Close down the piano-password.py script and open piano-decode.py using the Thonny IDE or the Text Editor. This script allows you to input your previously chosen musical password and decrypt the text stored in .pianovault

.



The script works similarly to the 'password' program in that it will ask you to key in your notes and display them as you type. The 'Instrument' key is used to confirm you've entered the password, at which point the procedure confirm_password is launched.
This procedure essentially works in reverse to encode_text. The script loads the hashed password and compares it to the notes you've input. If successful, it launches the procedure decode_text.

11 Review text decryption

The decode_text procedure firstly loads your encrypted data from .pianovault. If you've decided to change the location or name of the file holding your encoded text in

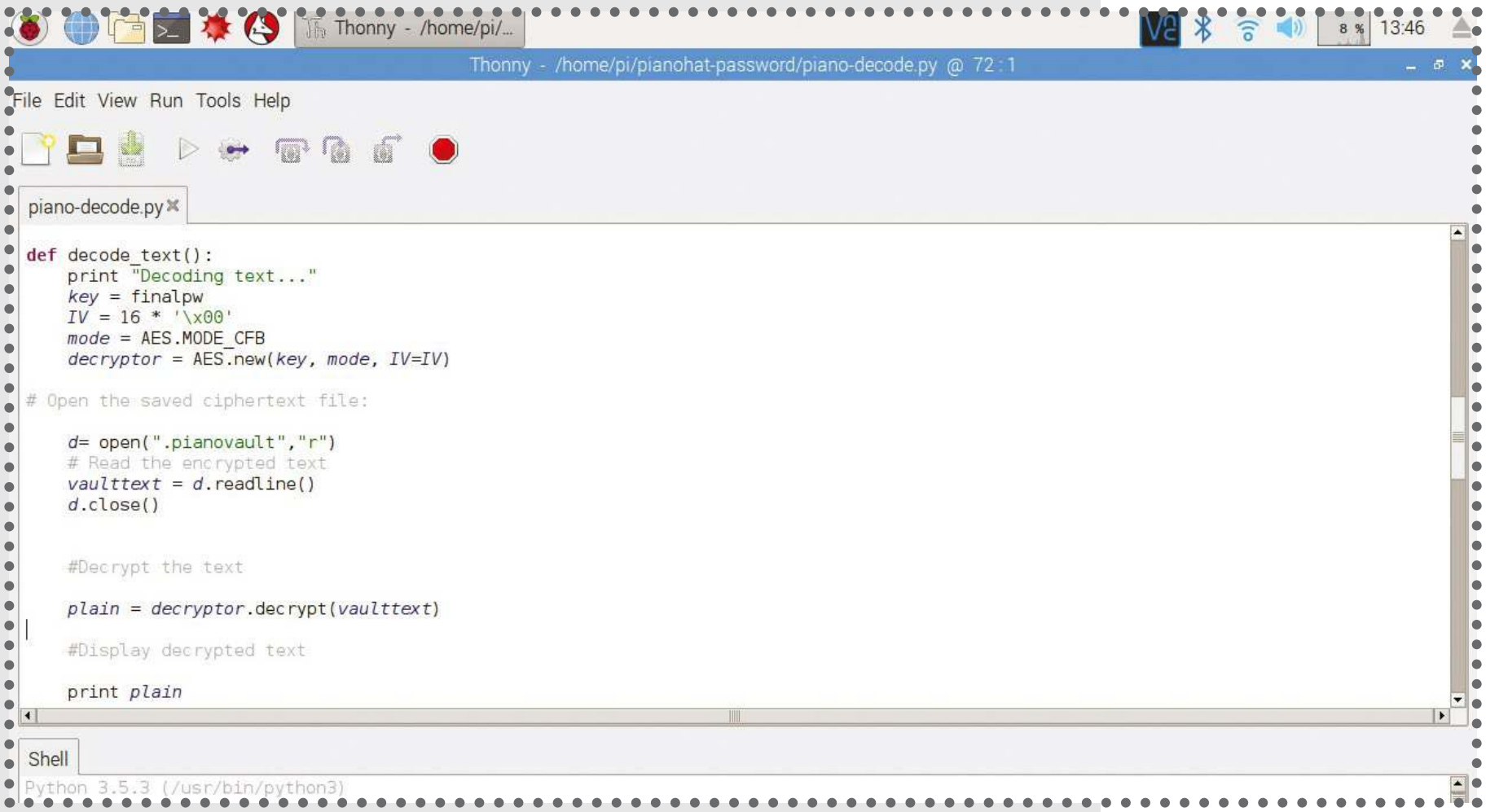
the piano-password script, make sure to change it here too.

The beauty of AES encryption is that it's symmetric – the same password used to encode data can be used to decode it. Here the procedure simply takes the key tones that you inputted earlier and displays them via the Terminal window.

Both the .pianohash and .pianovault files are opened as read-only to make sure the data isn't corrupted. If you want to make changes during runtime, change the option r to w+.

12 Create your first musical password

Having got your hands dirty with the Python code, exit Thonny or your Text Editor and open Terminal. Make sure your speaker or headphones are connected and run the piano-password script with:



```
  sudo python pianohat-password/piano-
password.py
```

The script will welcome you and ask you to use your notes. Remember, you can move up or down an octave using the buttons below the 'Instrument' button.
In the example, we've used the five-note motif from Close Encounters of the Third Kind: G, A, F, (octave lower) F, C. Ideally, you should use an original tune that you can remember easily. Make a note of the channel numbers on paper as a backup.

13 Encode your message

Press the 'Instrument' key when you have finished entering your tones. Your password will be printed to the Terminal as a series of numbers corresponding to button presses, as outlined above. The script will also play back the tones to you and will go on to save a hash of your password.

Now enter your secret text. There's no real limit on what this can be. Feel free to encode a password, a message for a friend, or a weblink to a non-text file such as an image.

When you have finished typing, simply press Return. The script will report the encrypted text has been saved.

14 Decode your message

When ready to retrieve your data, reopen Terminal on your Pi or connect via SSH and run:

```
sudo python pianohat-password/piano-decode.py
```

Play your musical password, then press the 'Instrument' button. If the password is incorrect, you'll see

a message to that effect and the script will exit.

If the password is correct, the script will display the message 'Password is Correct' and inform you it's in the process of decoding your text. The decrypted message itself will appear below.

While the decrypted text is displayed here in the Terminal, you cannot alter it. If you want to do so, run the piano-password.py script again.

15 Changing tones

Sharp-eyed readers will have noticed that in modifying the 'Instrument' button to enter passwords, you can now no longer choose between drum beats and piano tones. If you are more at home with a hi-hat than a high

A, you can switch to using drum sounds quite easily.

Open Terminal on your Pi and copy the drums directory from the Pimoroni examples folder into the pianohat-password folder:

```
cp -r /home/pi/Pimoroni/pianohat/examples/sounds/drums /home/pi/pianohat-password/sounds
```

Playing passwords

If you haven't played piano before, Pimoroni has put together an excellent tutorial for you. Open a Terminal and run:

```
sudo python Pimoroni/pianohat/examples/learn-to-play.py
```

Each of the keys on the Piano HAT has a built-in LED. Simply press the keys that light up to play a simple melody (in this case Twinkle, Twinkle, Little Star). The tutorial has the added advantage that if you accidentally press any other keys besides the one that is lit, no sound will play.

Make sure to move the piano folder outside the sounds folder too, e.g. by moving it to the trash. This ensures there's only one instrument for the script to choose.

16 Add to your scripts

When ready to retrieve your data, reopen Terminal on your Pi or connect via SSH and run:

```
sudo python pianohat-password/piano-decode.py
```

Play your musical password, then press the 'Instrument' button. If the password is incorrect, you'll see a message to that effect and the script will exit.

If the password is correct, the script will display the message 'Password is Correct' and inform you it's in the process of decoding your text. The decrypted message itself will appear below.

While the decrypted text is displayed here in the Terminal, you cannot alter it. If you want to do so, run the piano-password.py script again.

Playing passwords II

Once you feel comfortable with the basics, if you want to choose some random notes for your musical password, visit https://random.bretpimentel.com. Use the 'Options' button to increase the number of notes (up to 7). To generate a longer set of notes, visit www.random.org/integer-sets. This will allow you to generate a set of unique integers. You can use the same numbering system (1-60) as outlined in the scripts for notes, or make up your own.

Build with Minecraft

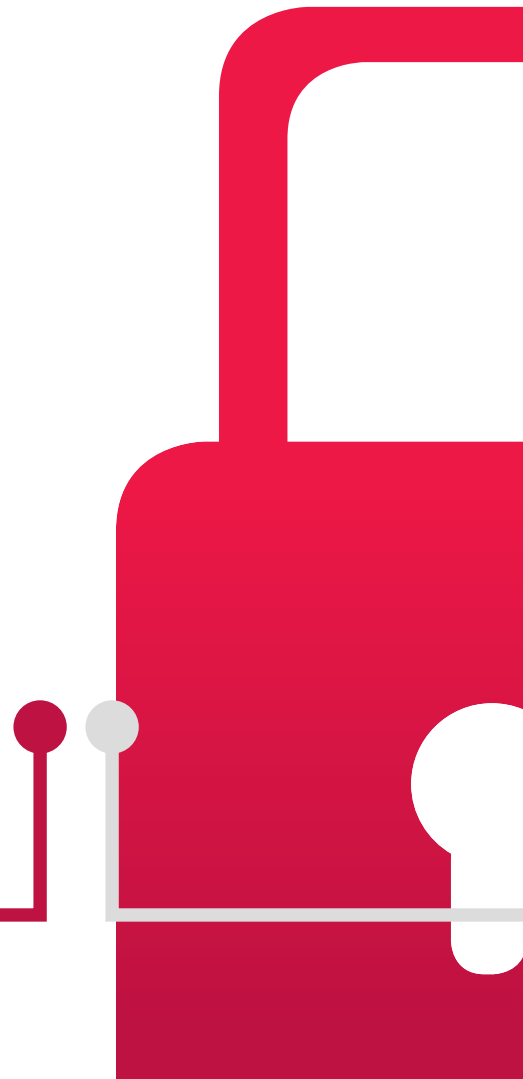
This issue, we will look at how to use Python to do really cool things within Minecraft on your Raspberry Pi

Minecraft has been a huge gaming environment for some time now. What many Minecrafters may not know is that there is a version of the Minecraft server that runs on Raspberry Pi. Even fewer may know that there is a Python engine buried down in there that you can write code for and get it doing some really weird and wonderful things. This issue, we'll look at how to get started writing Python code and interacting with your Minecraft world.

We'll assume you have a Pi 3 to work on for the rest of this article. The first step is to get Minecraft installed on your Raspberry Pi, although it's built in to the latest version of the Raspbian OS. To install Minecraft, use the command:

```
sudo apt-get install minecraft-pi
```

You can start up Minecraft from either the desktop Programming menu or a Terminal window with the minecraft-pi. command. You can then click on 'Start



Game' and then 'Create New' to get a new world up and ready. If you wanted to use Minecraft in the usual way, you could stop here and go start building in this new world. But we'll be programming it!

In order to write Python code within this new world you created, you'll need to open a Python console: open Programming > Python 3 IDLE).Let's start off with a simple Hello World program:

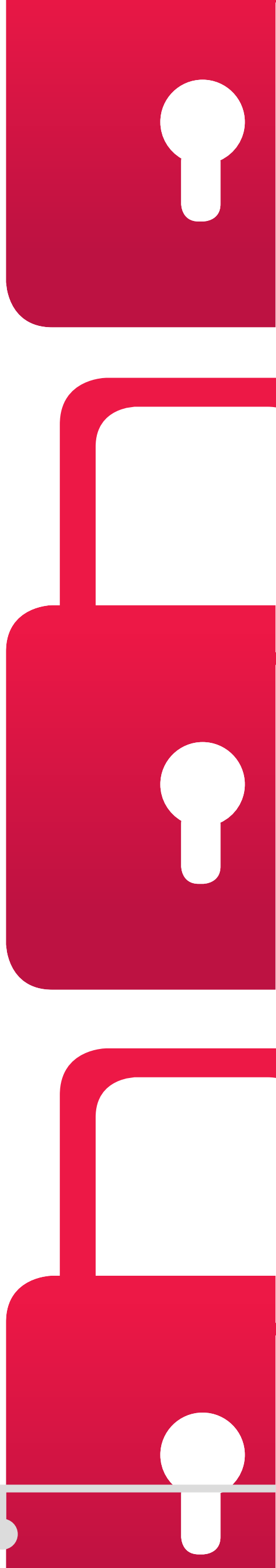
```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
mc.postToChat("Hello World")
```

The first step is to import the Minecraft class, and then instantiate a new object connected to the current game. The last line displays the text 'Hello World" in the chat window. If you are using the Shell console, each line of Python code gets executed as soon as you hit Enter. If you have a longer chunk of code you want to write, you can open a file to write your code into. Then, you can have the entire block executed when you save the file and then hit F5 to run it.

The first thing you will likely want to do is to interact with your player character within the game. You can get your character's position with code like the following:

```
curr_pos = mc.player.getPos()
```

The returned position object contains values for x, y and z. You could also do simultaneous assignment to three variables so that you get the three separate values immediately. Along with getting the current



position, you can set the position of your character with the following command:

```
x,y,z = mc.player.getPos()
mc.player.setPos(x+100, y, z)
```

This jumps the player character 100 spaces to the side. In the Minecraft world, the directions along the ground are labelled by x and z, whereas the y coordinate gives you a location up and down into the air. Be careful with that y coordinate. You may end up teleporting far up into the sky and then plummeting back down to the ground. Don't forget about gravity.

Along with the ability to move your character, you can interact with blocks which are what make up the world you are inhabiting. The most basic method available is the setBlock() method. For example, the following code would place a stone block beside you:

```
x,y,z = mc.getPos()
mc.setBlock(x+1, y, z+1, 1)
```

The parameters are the x, y and z locations values, along with an ID indicating what kind of block is to be created. The most common blocks are:

- 0 air
- 1 stone
- 2 grass
- 3 dirt

While you could use these raw ID numbers, there is a block class that provides an easier way to work with Minecraft blocks. You could do the same setBlock() command from above with the following code.

Why Python?

It's the official language of the Raspberry Pi. Read the docs at python.org/doc

```
from mcpi import block
mc.setBlock(x+1, y, z+1, block.STONE.id)
```

There are equivalent parameters for the other types of blocks available. There are also other options available for certain types of blocks. For example, for the wood block, you can set the type of wood from varieties like oak, spruce or birch, among others. You can set this with a fifth parameter to the setBlock() method. Sometimes, you may need to check to see what type of block is in some particular location. For example, you may need to check whether some piece of land will support flowers before trying to plant them; you could do it with the following code:

```
curr_block = mc.getBlock(x, y, z)
if curr_block == block.GRASS.id:
    mc.setBlock(x, y, z, block.FLOWER_CYAN)
```

Along with working with blocks one at a time, you can also work with a whole group of blocks together as one unit. For example, you can create an entire wall of ten stone blocks with the following code:

```
mc.setBlocks(x, y, z, x+10, y, z, block.STONE.
id)
```

You can do the reverse and use the method getBlocks(), with a starting point and end point, to get a list of block IDs for the blocks within that region. You can then go through each one and do whatever action you want to program.

Along with creating and placing blocks, you can also interact with them in a more direct manner. This is handled through events. An event occurs when a block is hit by one of the players within a given world. You can get the details for a given event through the following code.

```
curr_event = mc.events.pollBlockHits()
```

This event object contains several properties that you can access to see what is happening. The first property is the type of event. Currently, the only event implemented is BlockEvent.HIT.
The location of the block affected by the event is available from the .pos property. If it is important for the kind of interactivity that you want to code, you can even find out which face of the block was hit from the .face property. If you are in a multiplayer game, you can find out which player caused the event with the .entityId property. You can have a block only act if a particular player hits a particular block on a particular face. This can be put to great use in puzzle-type games.

"You can interact with blocks which make up the world"

To get a better look at all of the work you have been doing, you can play with the camera settings in order to change your viewport into your Minecraft world. You can use the following code to set the camera to a given location:

```
mc.camera.setPos(x, y, z)
```

If you happen to be in a multiplayer game and you want to have the camera follow one particular player, you can use the following command to do just that.

```
mc.camera.setFollow(playerID)
```

You'll have to find the ID of the player in question first. Once the camera is set, you can change the mode with the method mc.camera.setFixed(). You can reset it to the usual mode with the method mc.camera.setNormal(playerID).

Two final methods that will be very useful allow you to save and restart sessions. Before embarking on some extravagant test, you can use the following code to save a checkpoint:

```
mc.saveCheckpoint()
```

You can then reset your world to this checkpoint with the next command:

```
mc.restoreCheckpoint()
```

This way, you can really go wild with the ability to instantly undo any damage that you caused. Hopefully this article has whetted your appetite enough to dive into our Minecraft coding series.

Next issue

Get inspired **Expert advice** **Easy-to-follow guides**

"Probably the smallest, lightest PI tablet in existence"

Get this issue's source code at:
www.linuxuser.co.uk/raspicode